Praktische Aspekte der Informatik

Thomas Löwe
Prof. Marcus Magnor
Your Proposal

It’s due 15.05.2016!

It’s due 22.05.2016!
Software Versioning

SVN basics, workflow, and commands
Warning!

The following slides are meant to give you a very superficial introduction.

If you want to learn more, have a look at:

http://subversion.apache.org
http://svnbook.red-bean.com
http://tortoisesvn.net
• What is versioning?
  ▪ Naïve approaches
  ▪ Smart approaches

• SVN Workflow
  ▪ Basic svn commands

• Assignment: Check in your proposal
A simple file server

Two users read the same file: Harry and Sally both read a file A from a repository.

They both begin to edit their copies: Harry and Sally each make a copy A' and A'' of file A.

Harry publishes his version first: Harry writes his version A' back to the repository, and Sally tries to write her version A'' by overwriting Harry's A' in the repository, which causes conflicts.

Sally accidentally overwrites Harry's version: Sally's attempt to overwrite Harry's version A' results in a conflict, and Sally has to resolve the version conflict.
Lock-modify-unlock

Harry "locks" file A, then copies it for editing

Repository

Harry

A

LOCK

Read

Sally

A

While Harry edits, Sally’s lock attempt fails

Repository

Harry

A

Sally

A

Lock

Harry writes his version, then releases his lock

Repository

Harry

A'

Write

UNLOCK

Sally

A

Now Sally can lock, read, and edit the latest version

Repository

Repository

Harry

Sally

A'

Read

LOCK

A'

A'
Copy-modify-merge

Two users copy the same file

Repository

A

Read

Read

Harry

A

Sally

A

They both begin to edit their copies

Repository

A

Harry

A'

Sally

A''

Sally publishes her version first

Repository

A''

Write

Harry

A'

Sally

A''

Harry gets an “out-of-date” error

Repository

A''

Write

Harry

A'

Sally

A''
Copy-modify-merge

Harry compares the latest version to his own

Repository

Read

A''

Harry

Sally

A''

A’

A''

A'

A''

Harry

Sally

A''

A'

A''

A'

A''

Repository

Write

Read

Now both users have each others' changes

The merged version is published

Repository

A'

Harry

Sally

A'

A''

A'

A''

Read

A''

Harry

Sally

A'

A''
• **Benefits:**
  - Global revision numbers
  - No serialization (parallel editing)
  - Offline editing
  - Merging works most of the time

• **Drawbacks:**
  - Requires a disciplined workflow and maintenance!
Initial checkout, and *regular update* to the newest version:

```
svn checkout / svn update
```

Make changes:

```
svn add / svn delete / svn copy / svn move / ...
```

Examine your changes:

```
svn status / svn diff
```

Possibly undo some changes:

```
svn revert
```

Commit changes:

```
svn commit
```
Always use the `update` command to get the latest version:

```bash
$ svn update
```

Basic Usage
19
U foo.c
U bar.c
Updated to revision 2.

For bug hunting it is useful to look at old revisions by adding

```
--revision [revision-number]
```
Check the status of all local files compared to the repository:

```shell
$ svn status
?
  local_settings.h # file is not under version control
A
  food/sandwich.h  # file is scheduled for addition
C
  food/sandwich.txt # file has conflicts from an update
D
  bread.cpp        # file is scheduled for deletion
M
  food/sandwich.cpp # the content has local modifications
```
$ svn log

Added includes and corrected the number of cheese slices.

$ svn diff
Index: sandwich.cpp

--- sandwich.cpp (revision 3)
+++ sandwich.cpp (working copy)
@@ -1,7 +1,12 @@
+#include <sys/types.h>
+#include <sys/stat.h>
+#include <unistd.h>
+#include <stdio.h>
int main(void) {
- printf("Sixty-four slices of American Cheese...\n");
+ printf("Sixty-five slices of American Cheese...\n");
  return 0;
If there is a conflict, all relevant versions are saved on disk:

```bash
$ ls -l
sandwich.txt
sandwich.txt.mine  # Your work
sandwich.txt.r1    # Old version
sandwich.txt.r2    # Someone else's work
```
$ cat sandwich.txt
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<<<<< .mine
Salami
Mortadella
Prosciutto
========
Sauerkraut
Grilled Chicken
>>>>>>> .r2
Creole Mustard
Bottom piece of bread
SVN Conflicts

You have the following options:

```
$ svn resolve --accept theirs-full sandwich.txt
```
Replaces your file with the current revision.

```
$ svn resolve --accept working sandwich.txt
```
You have corrected the conflict by hand.

```
$ svn resolve --accept mine-full sandwich.txt
```
Replaces their file with your current version.

```
$ svn resolve --accept base sandwich.txt
```
Reverts to the last revision you checked out (in this example, r1).

You should also clean up the temporary files *.r* and *.mine afterwards.
You can then commit your changes:

```
$ svn commit -m "Added more delicious ingredients."
Sending sandwich.txt
Transmitting file data.
Committed revision 3.
```
Things to watch out for

- Commit often
- Always use understandable messages
- Only commit code that compiles
- Avoid conflicts early, by designating responsibilities.

- Do not commit temporary files!
- Do not commit ‘backup’ files!
A. Commit your project proposal
Check out your project's directory:

```bash
svn checkout svn://elara.cg.cs.tu-bs.de/public/padi/yourname/
```

Copy your proposal (proposal.txt) into this directory. Make sure your proposal is a text file and that it contains both your name as well as your matrikelnummer.

B. Modify the projects.txt list
Check out the common directory:

```bash
svn checkout svn://elara.cg.cs.tu-bs.de/public/padi/common/
```

There should be a file called projects.txt. It contains a list of all projects and current states. Please add your project information to the file, so I know who is still working on his/her project.
Hallo,

wir wollen diese Woche am Institut für Computergraphik ein neues Verfahren zur virtuellen Objektbetrachtung testen und mit bestehenden Verfahren vergleichen.

Hierfür brauchen wir noch Probanden, die für uns einfache kleine Aufgaben mit einer vorher zufällig bestimmten Eingabemethode lösen (Maus&Tastatur, Touchscreen, Virtual Reality, ...). Die Durchführung des Tests wird pro Person ca. 30 Minuten dauern.

Wer von euch interessiert ist, kann sich hier für einen Termin eintragen: https://docs.google.com/spreadsheets/d/1kv7_bWL_CDZQnDLb0lp8k9SXXdwiE7wdh1xUbSKvbjM

Ihr findet uns dann im Raum G25 im Informatikzentrum (IZ) beim Institut für Computergraphik.

Wir freuen uns auf eure Teilnahme!